CẤU TRÚC DỮ LIỆU CÁC TẬP HỢP RỜI NHAU

(DSU – DISJOINT SET UNION / UNION FIND)

- Sử dụng làm gì? Gộp 2 phần tử trong 1 tập hợp lớn vào 1 tập hợp nhỏ, hoặc chỉ ra 1 phần tử có thuộc tập hợp không

- Coi mỗi phần tử là 1 đỉnh

- Sử dụng cấu trúc dữ liệu này để code Kruskal Algorithm

- Có 2 thao tác chính:

+ Gộp 2 tập hợp với nhau

+ Chỉ ra tập hợp nào đó sẽ chứa các phần tử như thế nào

- Các thao tác:

+ make\_set(a): tạo một tập hợp chứa phần tử a

+ Union (a, b): gộp 2 tập a và b

+ Find(a): tìm ra phần tử đại diện của tập hợp mà a thuộc về

A whiteboard with writing on it

Description automatically generated

I . MAKE \_ SET

- Cần đại diện cho tập hợp => Mảng parent[] lưu trữ duy trì đại diện của 1 tập hợp nào đó đề truy cập đại diện vs O(1)

A white board with writing on it

Description automatically generated

II. FIND()

- Một đỉnh nào đó có parent là chính nó thì đỉnh đó là đại diện cho tập hợp nó thuộc về

A white board with writing on it

Description automatically generated

III. UNION()

- Gộp 2 tập hợp chứa a và chứa b

- TH1: a và b đã thuộc cùng 1 tập hợp ⬄ parent[a] = parent[b]

TH2: a và b chưa cùng thuộc 1 tập hợp => cho parent[b] = a hoặc parent[a] = b;

A white board with blue writing

Description automatically generated

IV. Ví Dụ:

Cho

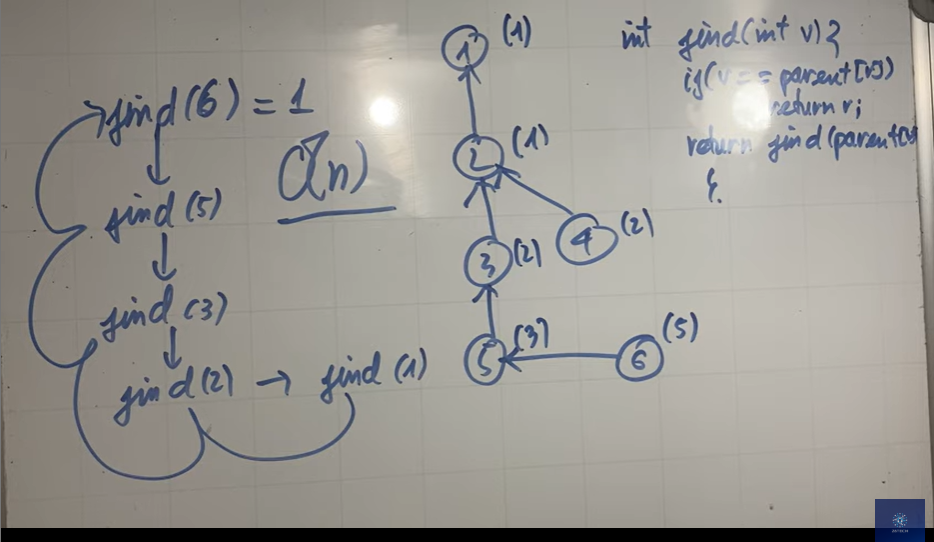
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| Sau khi make\_set(), parent: | 1 | 2 | 3 | 4 | 5 |
| Union(1,2)  Union(3,4) | 1 | 1 | 3 | 3 | 5 |
| Union(1,3) | 1 | 1 | 1 | 3 | 5 |
| Union(4,5) | 1 | 1 | 1 | 3 | 1 |

A white board with blue writing on it

Description automatically generated

- Một số vấn đề:

+ Một số trường hợp hàm FIND() gọi chuỗi đệ quy rất dài, mất O(n) để tìm đại diện



=> Cần phải tối ưu

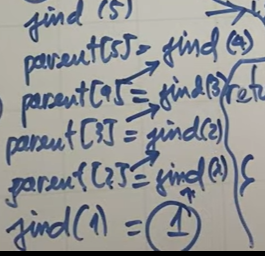
A whiteboard with blue marker on it

Description automatically generated

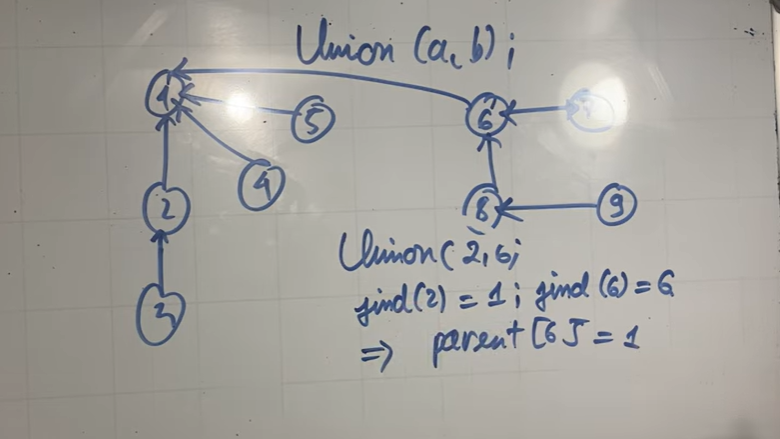
- Path compression: Nén đường áp dụng vào hàm FIND()

A white board with blue writing

Description automatically generated



- Tối ưu Union(a, b) bằng rank, size của tập hợp đó:



- Tập hợp nào size nhỏ hơn tức ít đỉnh hơn => số đỉnh bị thay đổi parent ít hơn

=> Size nhỏ hơn sẽ phải gọi size lớn hơn là cha

=> Cần thêm mảng size() để lưu size()

